



Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Performance Portability of Remote I/O in Distributed Workflows

NATHAN R. TALLENT, RYAN D. FRIESE, JOSHUA SUETTERLEIN, JAN STRUBE

Pacific Northwest National Lab

P3HPC Forum

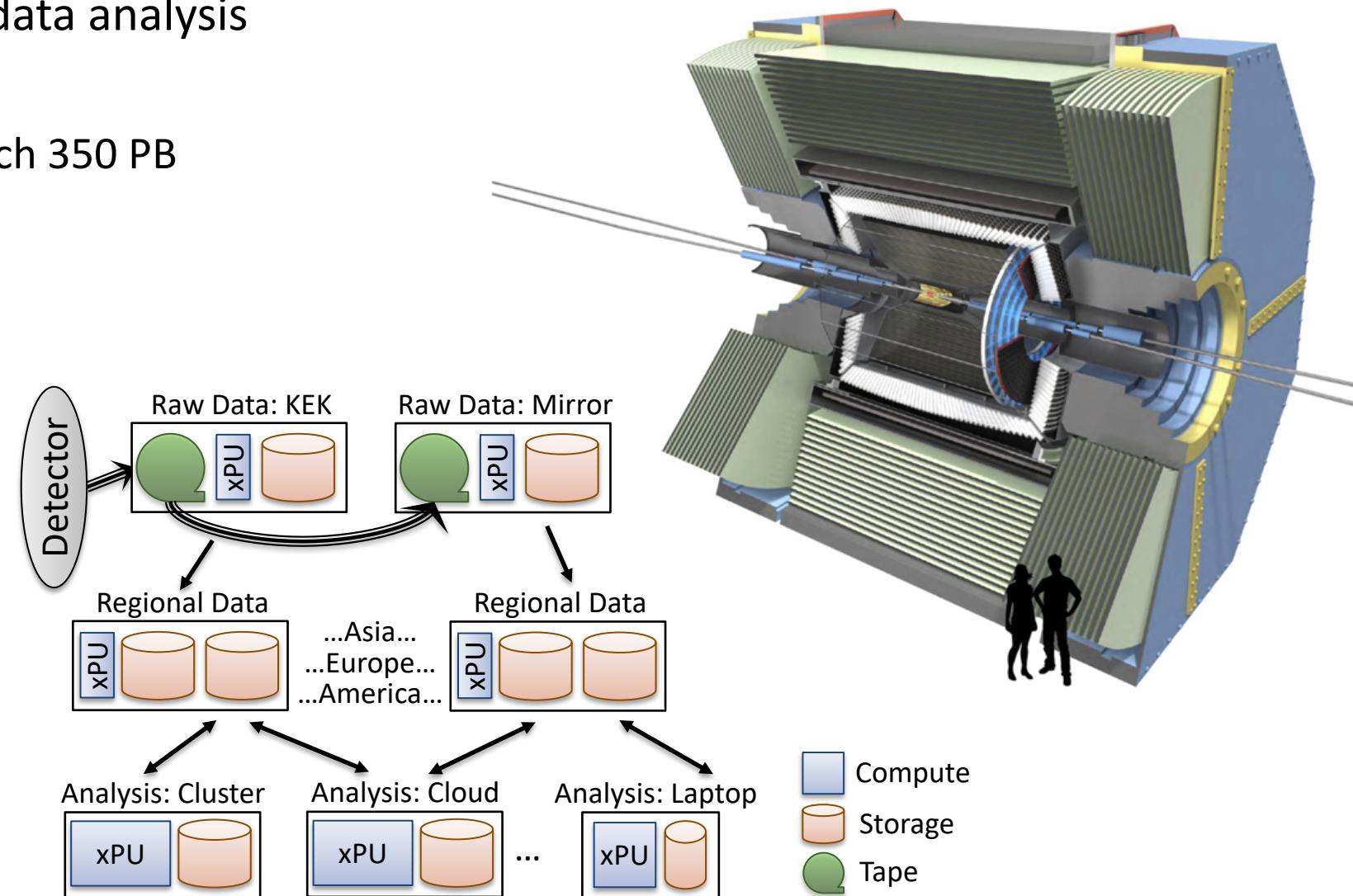
September 1, 2020

Scientific Workflow Analytics: Distributed, Data-intensive



- Belle II Workflow: Extensive data analysis
- Data! 25 PB/year of raw data
 - Stored data expected to reach 350 PB
- Big Data + Big Compute
 - Data movement & storage
 - Large working sets
 - Non-streaming accesses
 - Heterogeneous hardware

Data movement
Resource contention
Response time



Observation, Analysis & Acceleration for Workflows



Challenge

- Data movement bottlenecks
- Heterogenous hardware

★ Today's talk

Observe

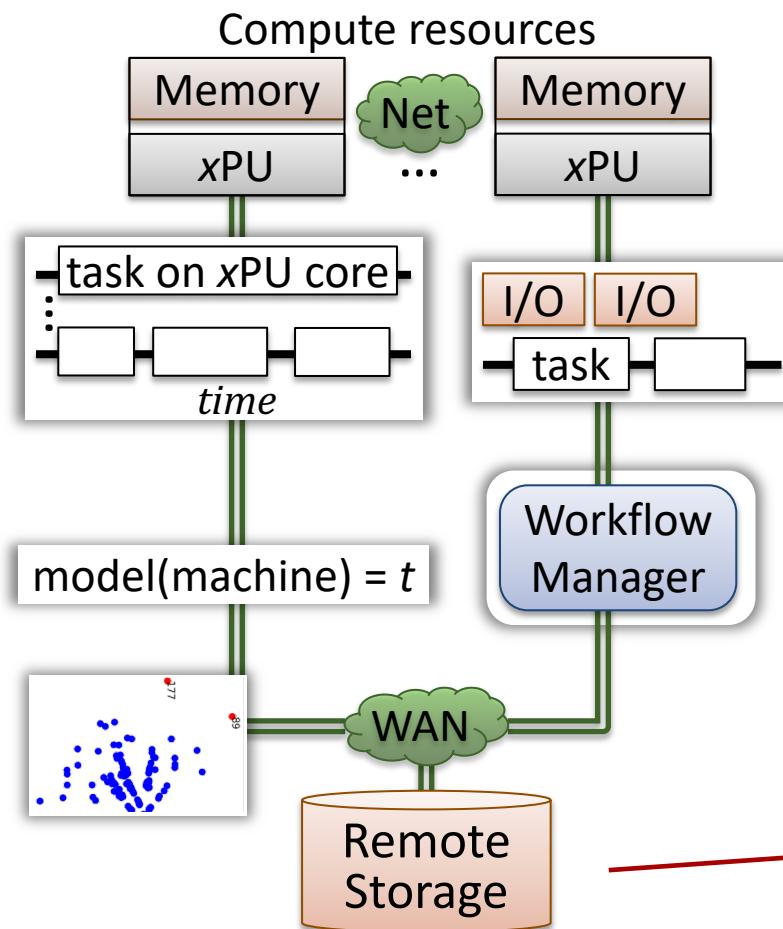
Monitor performance costs, task constraints, data movement

Analyze

Task models predict performance, to enable scheduling

Analyze

Detect anomalies during runtime, to proactively guide execution



Accelerate

Intelligent data movement to eliminate or hide remote I/O access costs

Accelerate

Scheduling evens I/O load and reduces response time across different hardware

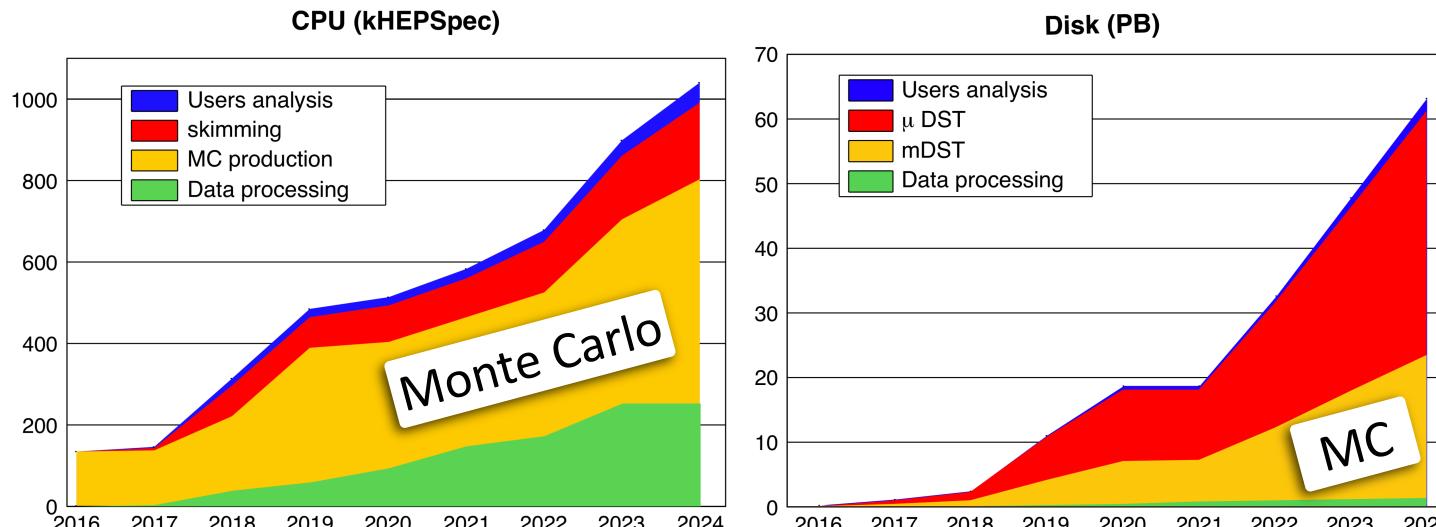
Accelerate

Dynamically move files for best throughput under different workloads

Belle II MC-Sim: Geographically Distributed Analytics

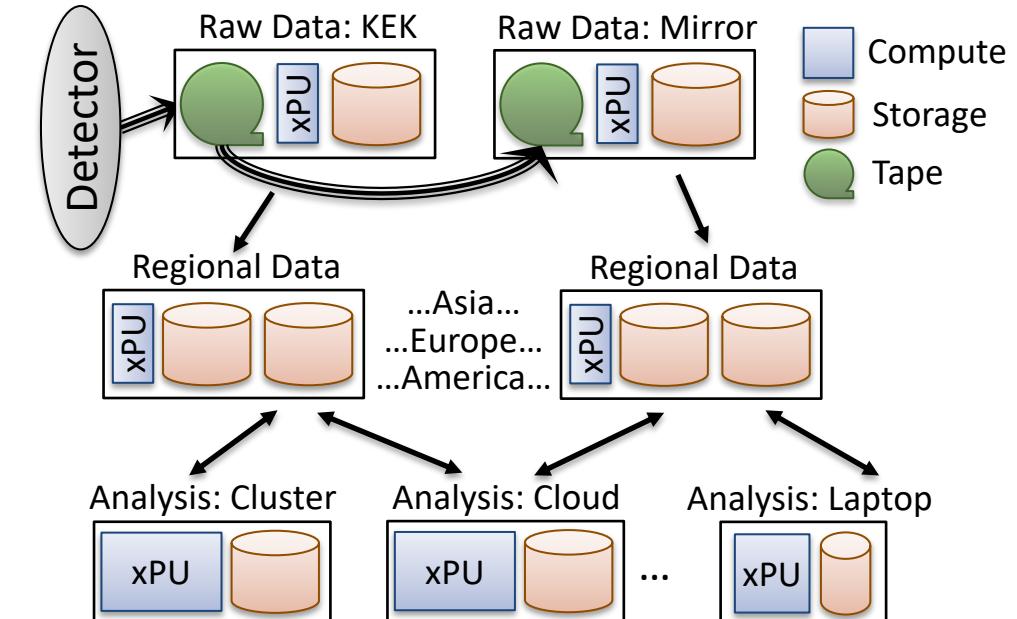


- Belle II Workflow: Extensive data analysis
- Monte Carlo Sim: Big Data + Big Compute
 - Many logically independent tasks
 - Read intensive: inputs >> outputs
 - Simple data/file consistency: no read/write mode
 - Non-streaming accesses
 - Some inputs may be common



Our experimental configuration:

- Each task: 16 KiB remote data per 2.5 ms
- Campaign: 48 Gb/s (6.0 GB/s)
- Two clusters: two 1 Gb/s WAN links
- Five node types
- Four remote sites: 2x PNNL, NERSC, ORNL



TAZeR: Transparent Asynchronous Zero-copy Remote I/O

github.com/pnnl/tazer

~~PNNL~~

Challenge: Remote I/O blocks task, consumes space, bottlenecks data movement

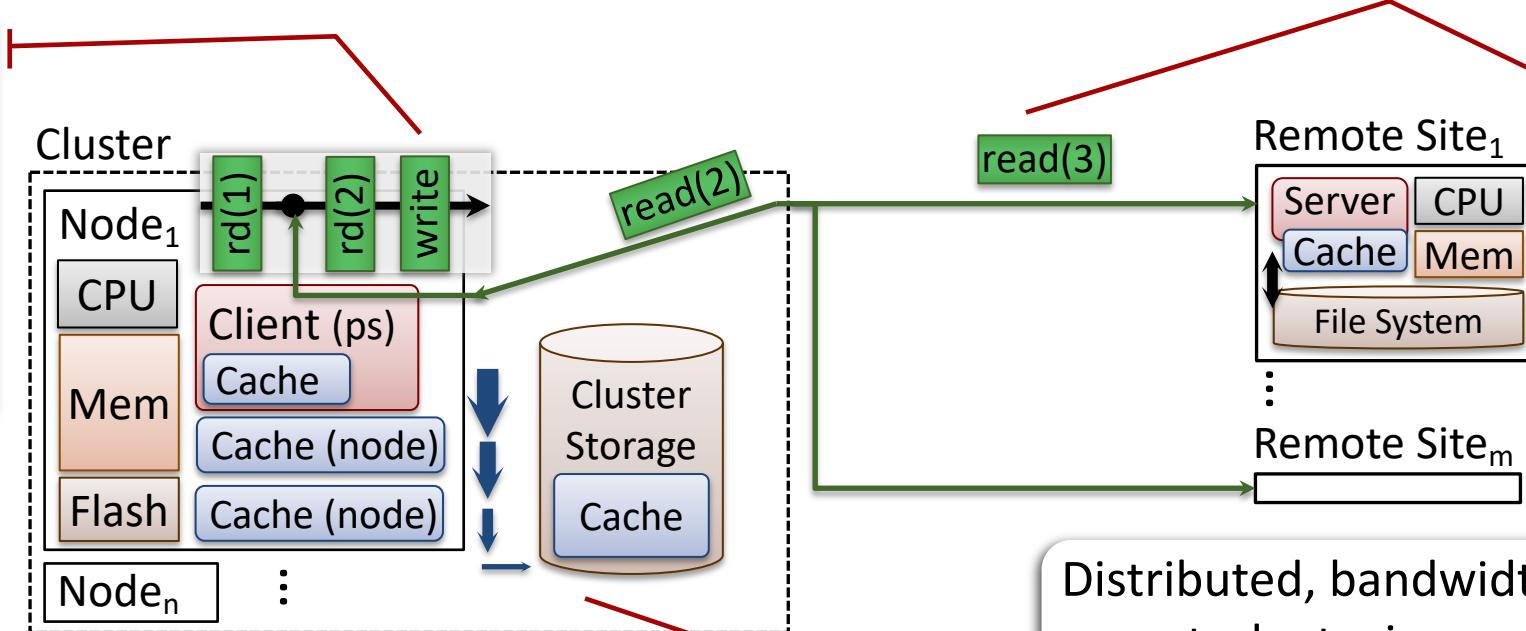
TAZeR...

- Transparent
- Increases reuse

- Increases scaling
- Reduces blocking
- Increases bandwidth

Overlap I/O & task

- read prefetching
- write buffering



Soft zero copy transfer

- directly to memory
- ensure minimum transfer latency

Distributed, bandwidth-aware staging

- controls staging capacity
- captures inter-task locality (spatial & temporal)
- asynchronous, bandwidth-aware updates
- exploits emerging storage technology
- scales (introduces no centralization)

Vary Methods for Remote I/O: Overview

Suettlein, Friese, Tallent, and Schram, "TAZeR: Hiding the cost of remote I/O in distributed scientific workflows" BigData 2019



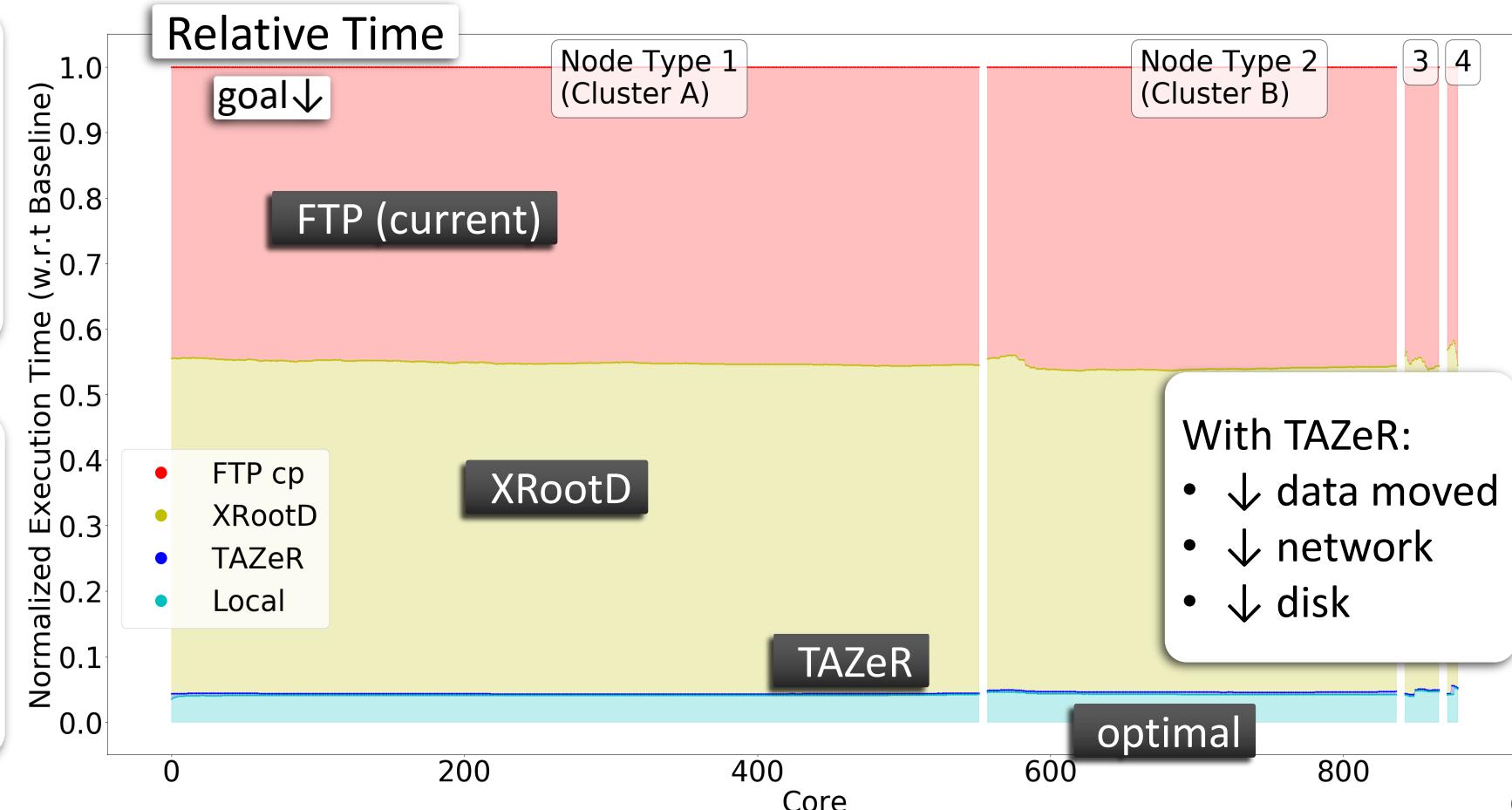
- FTP – current
 - Copy input files (24) to local disk
 - Transfer all data
 - Blocking

- Local – “optimal”
 - Data already present

- TAZeR: 22×/12× faster than FTP/XRootD
 - within 7% of optimal

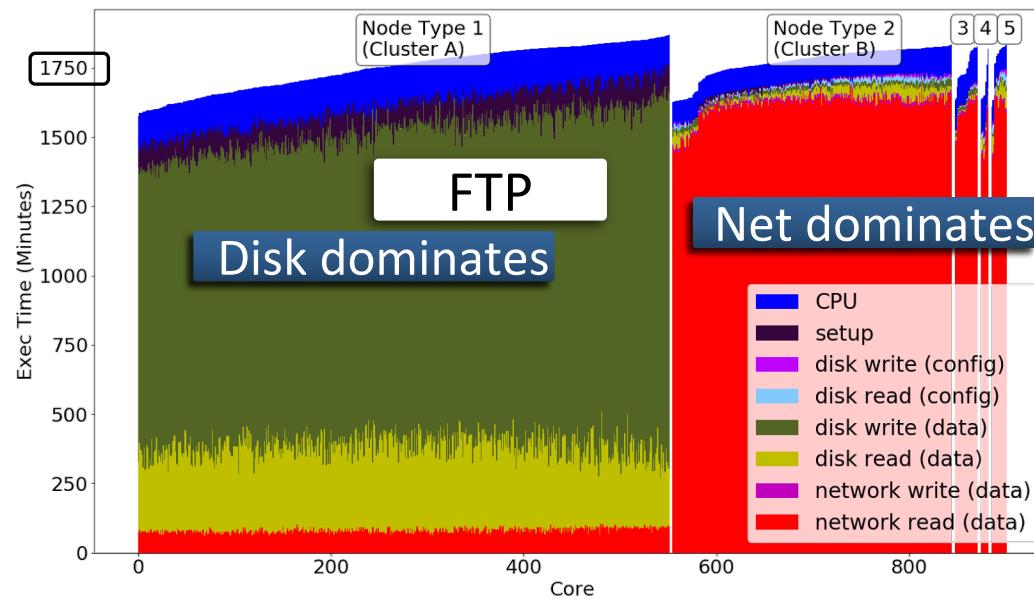
- XRootD: State of the art
 - per task staging (1-4GiB)
 - Unbounded cluster filesystem staging
 - Same remote sites
 - Version 4.9.1

- TAZeR
 - L1 – 64MiB private mem
 - L2 – 16GiB shared mem
 - L3 – unbounded cluster filesystems
 - L4 – remote data sites
 - 1MiB-sized blocks



Vary Methods for Remote I/O: Execution Time

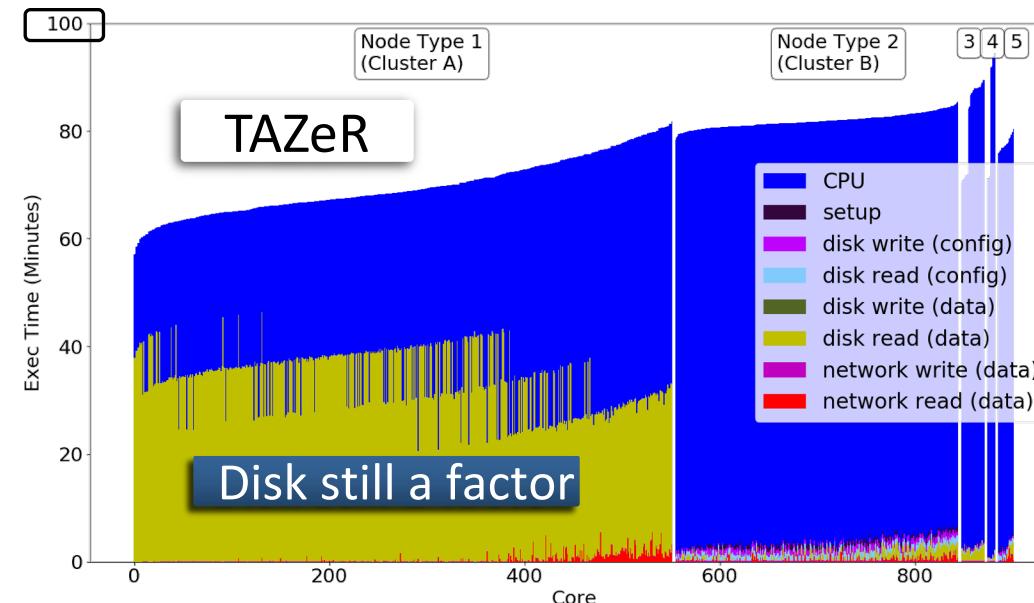
PNNL



goal ↓

TAZeR is

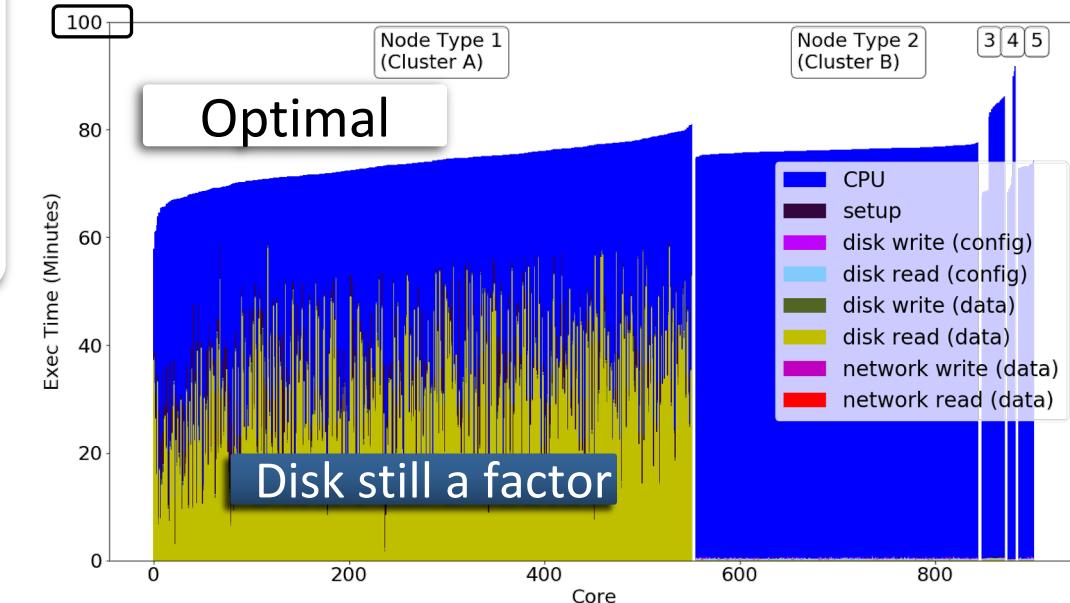
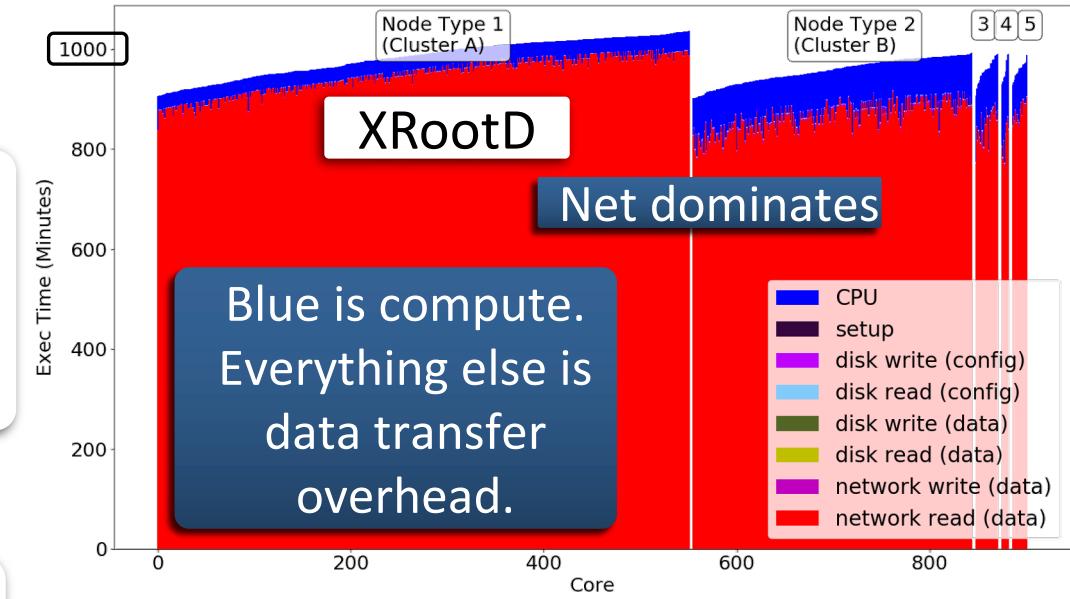
- 22x < FTP
- 12x < XRootD
- 7% > Optimal



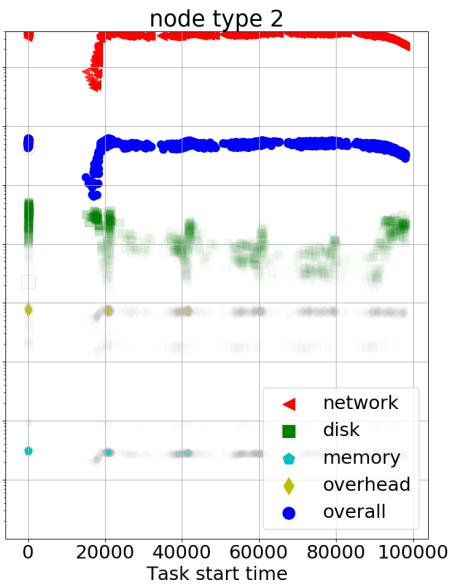
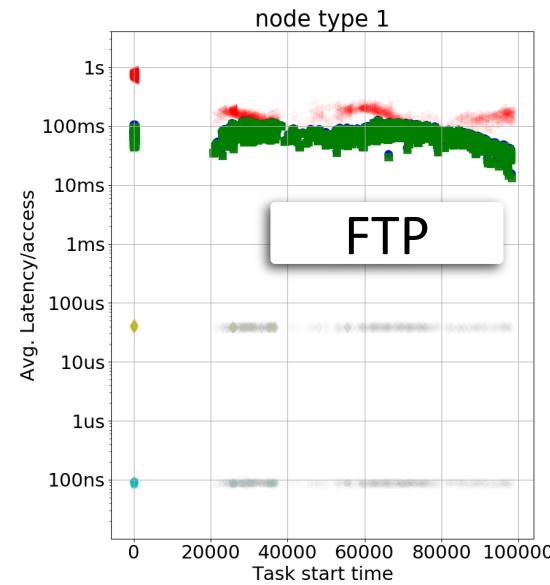
With TAZeR:

- ↓ moves
- ↓ NW
- ↓ disk

Feedback to
Belle II/KEK

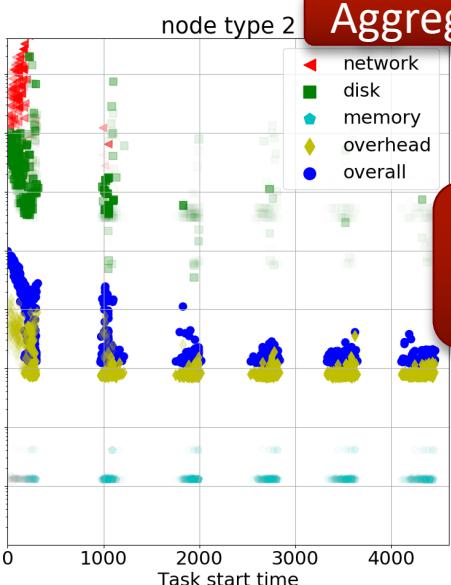
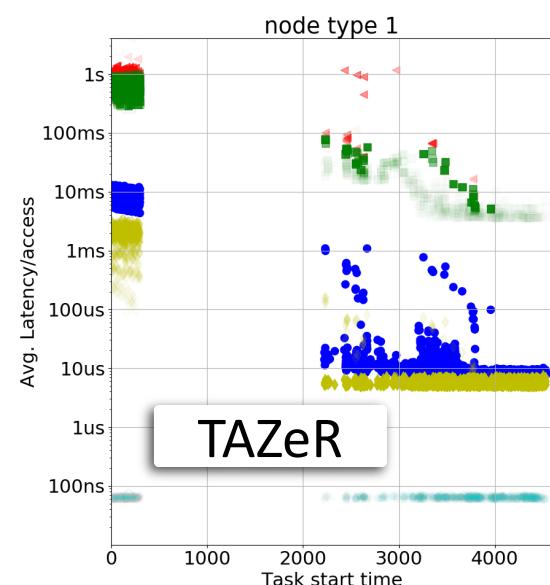


Vary Methods for Remote I/O: Breakdown of Read Time



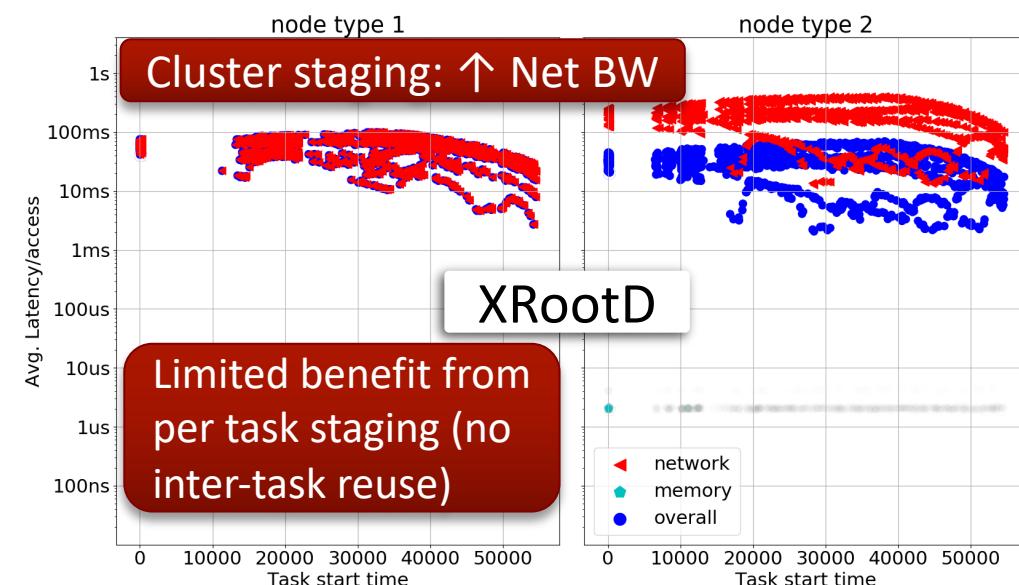
goal ↓

Blue is average access time for task I/O
Other colors (and intensity) is percent a task waits per resource

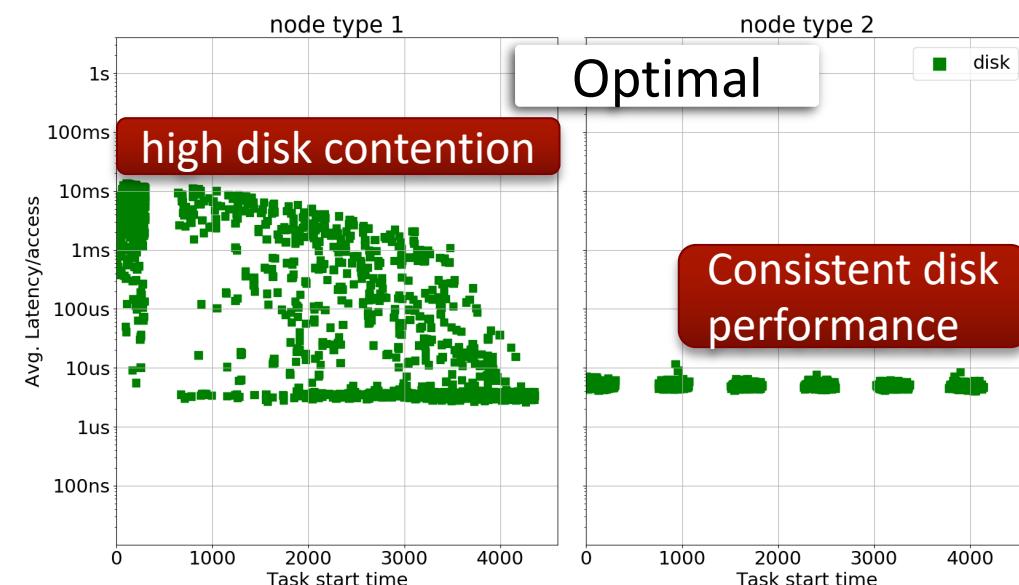


Aggregation: ↑ Net BW

Staging: ↓ accesses to slow resources (high inter-task reuse)



Limited benefit from per task staging (no inter-task reuse)



Vary Ratio of Staging to Footprint



What happens when reuse changes?

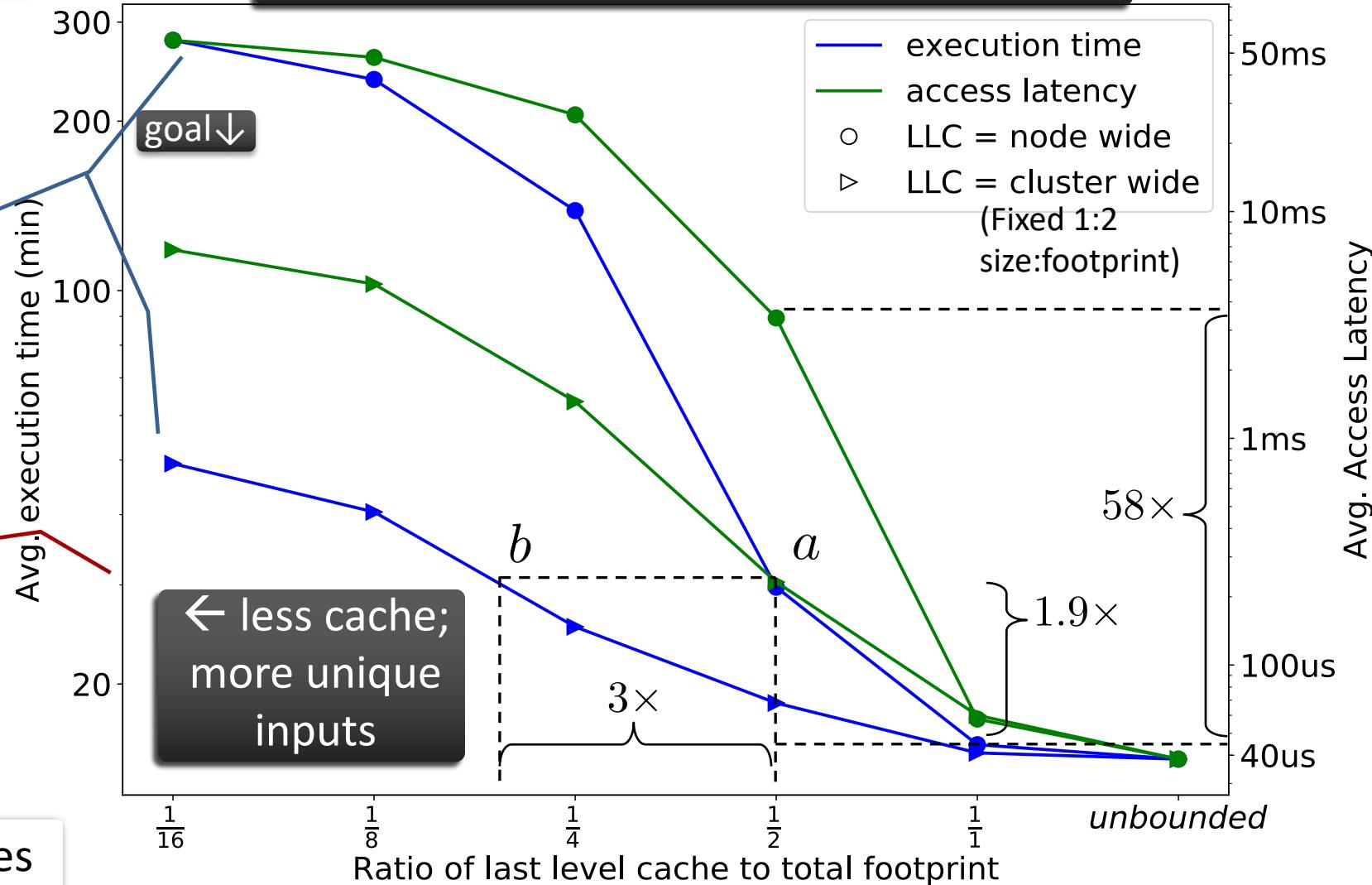
TAZeR exploited read reuse
• 4K tasks, 5 BG sets

Cluster-shared (v. node-shared)
• gentler slope
• fewer network transfers
• extracts more reuse

Cluster-shared iso-performance
($a \rightarrow b$) with node-shared, but:
• 3x smaller staging area
• 3x more unique inputs

Next: adaptivity; scaling mutexes

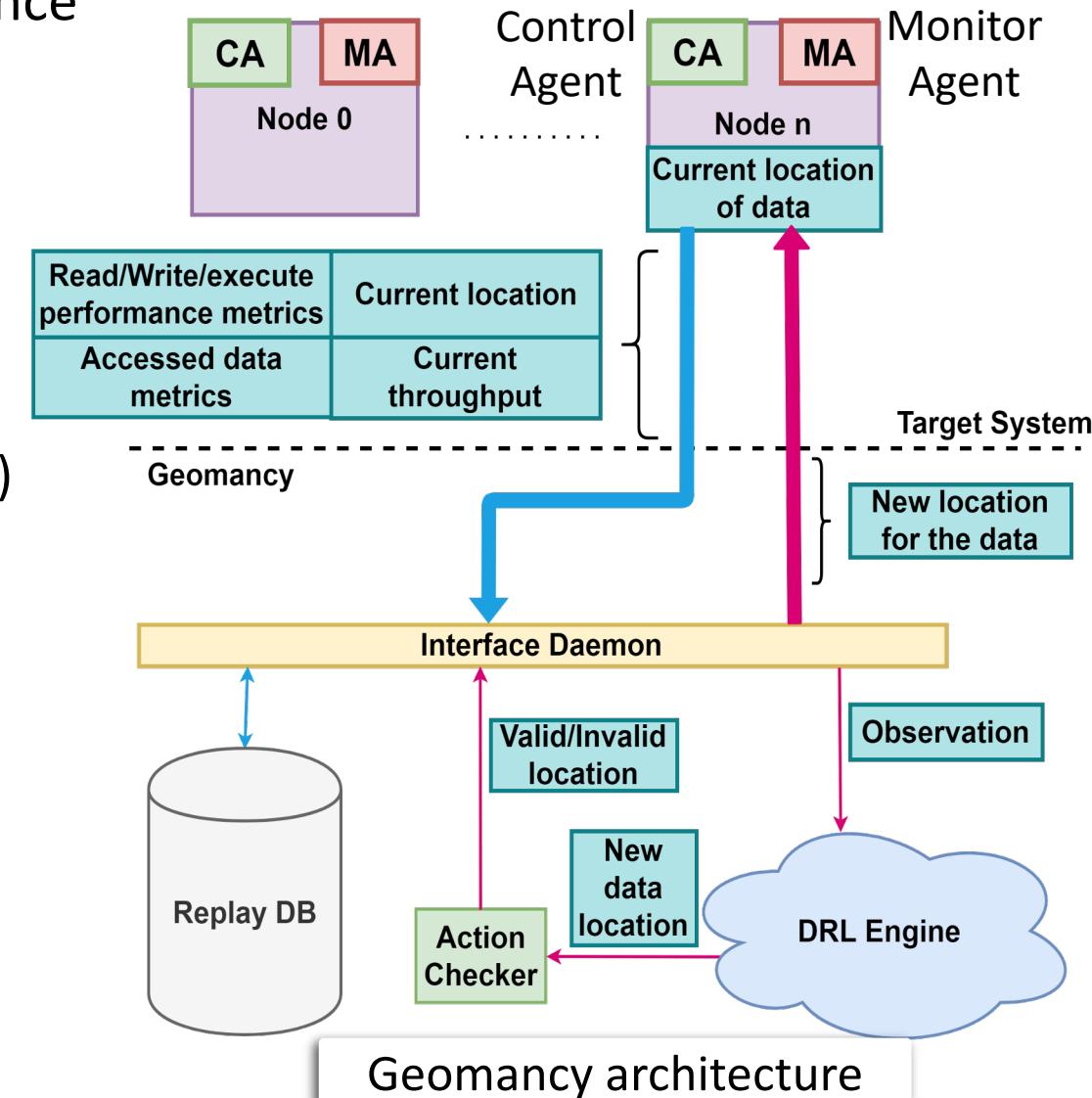
Vary ratio of cache to total footprint \approx
Vary ratio of tasks to unique input sets



Geomancy: Adjust Data Layout To Improve Throughput

O. Bel, et al. "Geomancy: Automated Performance Enhancement through Data Layout Optimization" MSST 2020.

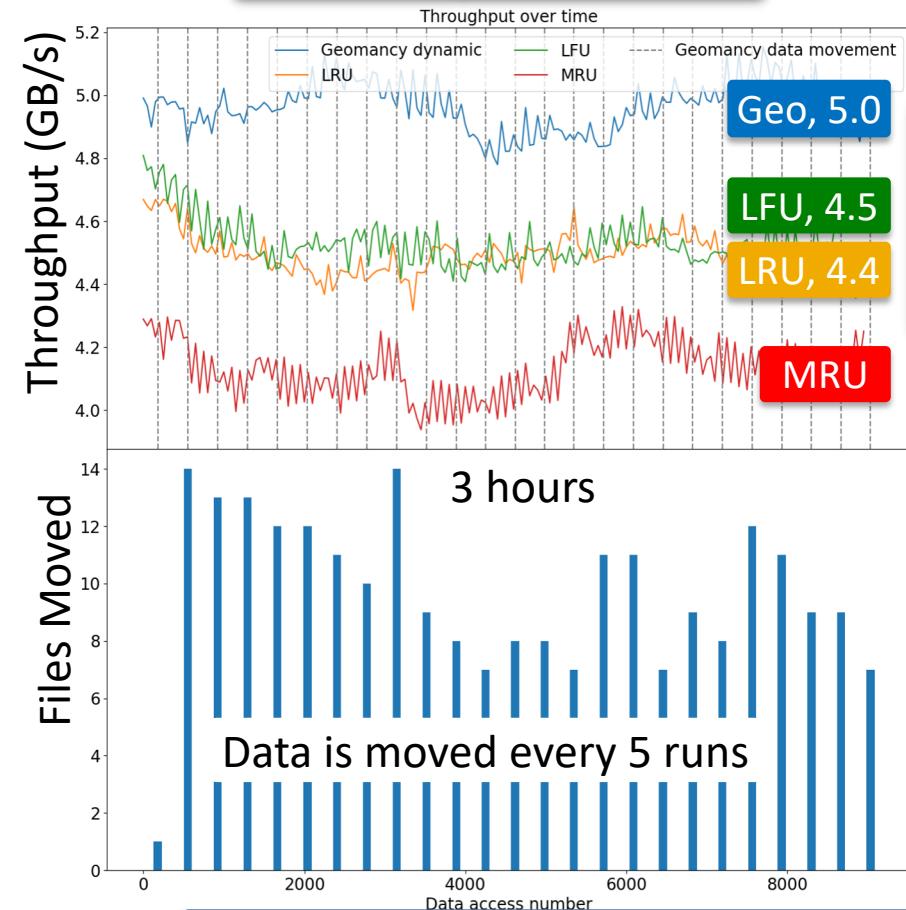
- Data layout (storage) significantly affects performance
 - layout = map of files to storage system/devices
- Adjusting data layout can
 - tailor mapping for varying workloads
 - I/O access model, intensity, access patterns
 - avoid congestion from other workloads
 - avoid failures (storage) & misconfigurations (nw link)
- Geomancy uses reinforcement learning
 - monitor I/O activity and data layout
 - model performance (throughput)
 - predict performance during next window
 - select and realize new layout



Comparing Geomancy with Layout Heuristics



Geomancy v. Heuristics



Geomancy learns:

- indicators of read contention
- RAID-5 has different read/write speeds

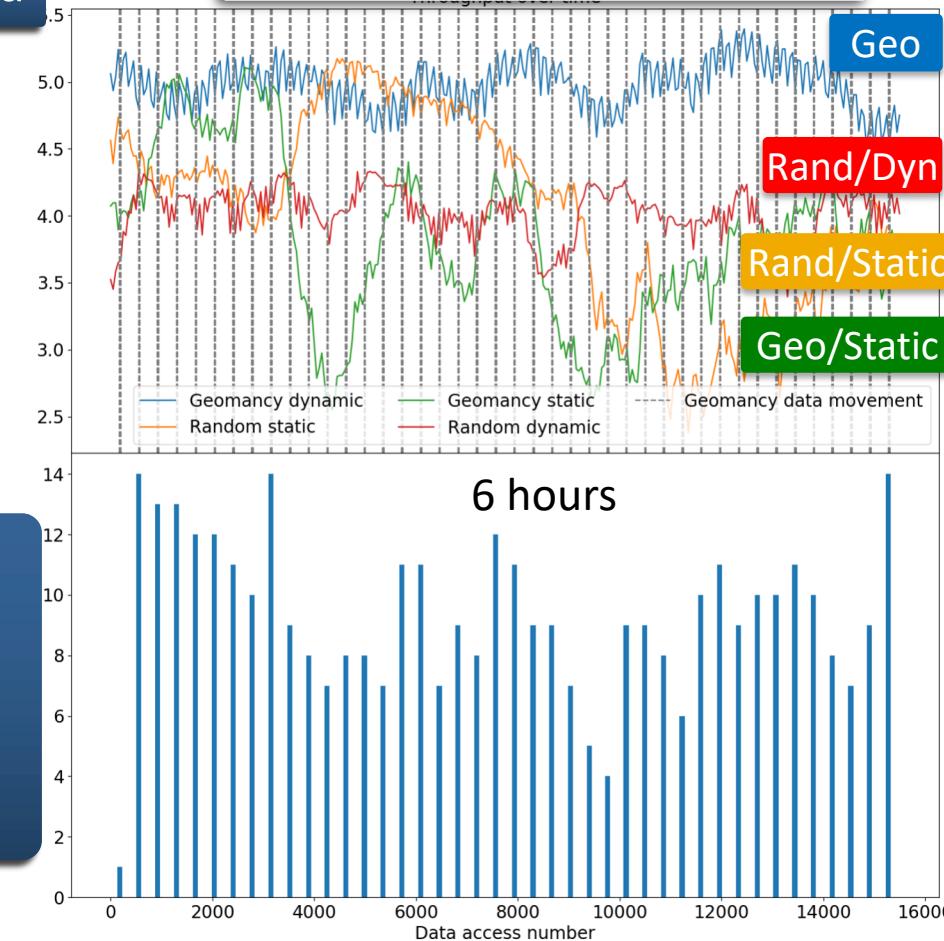
Belle II MC Workload

LFU: Least frequently used...
LRU: Least recently used...
MRU: Most recently used...
file to slowest

Geomancy: 11% over LFU

- Geomancy v. Rand/Dyn
- >20%
- Geomancy v. Static
- higher performance
 - lower variation

Geomancy v. Baselines



Adjusting layouts dynamically results in more consistent and higher performance

What performance is possible when varying I/O parameters?



$\mathcal{I}_{\text{rate}}$ Intensity
Change I/O per work
• Data reducers
• Work aggregation

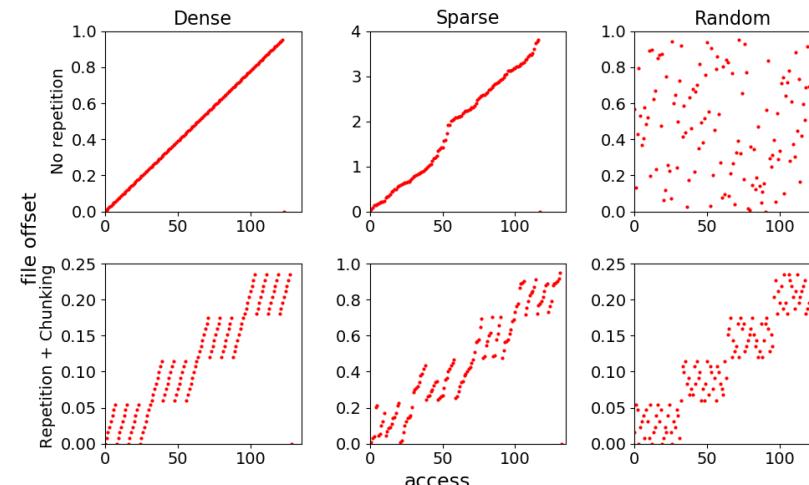
\mathcal{I}_{use} Data Uses
Tasks reuse data for write-once read-many

\mathcal{I}_{loc} Locality-aware
Ensembles & chunking for temporal I/O locality
• I/O-aware access/schedules
• In-situ computation

\mathcal{I}_{pat} Access Pattern
• Data layout for I/O spatial locality
• Regular accesses for prefetching

Parameter	Possible values					
$\mathcal{I}_{\text{rate}}$ (MB/s) (vs. WAN)	125 (1x)	250 (2x)	500* (4x)	1000 (8x)	2000 (16x)	
\mathcal{I}_{use}	2	4	8	16	64	128* 256
$\mathcal{I}_{\text{loc}-\epsilon}$	1*			4		16
$\mathcal{I}_{\text{loc}-c}$		whole file				64 MB
\mathcal{I}_{pat}	(ordered) dense*		(ordered) sparse		random (dense)	
$\mathcal{I}_{\text{pat}-\delta}$	1.0		0.25		1.0	
\mathcal{I}_{rt}	Baseline		Optimal		TAZeR	

\mathcal{I}_{rt} Runtime
• Baseline:
copy in/out
• Optimal:
pre-staged



In progress